(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開平8－152881
(43)公開日　平成８年(1996)６月11日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ１０Ｈ　1/00	１０２　Ｚ			
	Ｚ			
Ｇ１０Ｋ　15/04	３０２　Ｄ			

審査請求　未請求　請求項の数５　ＦＤ　（全９頁）

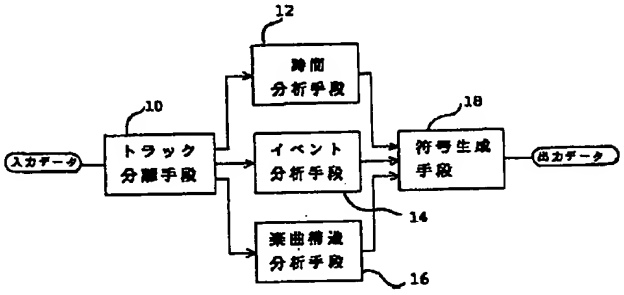
(21)出願番号　特願平6－319337

(22)出願日　平成６年(1994)11月29日

(71)出願人　000004329
日本ビクター株式会社
神奈川県横浜市神奈川区守屋町３丁目12番地
(72)発明者　宍戸　一郎
神奈川県横浜市神奈川区守屋町３丁目12番地　日本ビクター株式会社内
(74)代理人　弁理士　二瓶　正敬

(54)【発明の名称】　演奏情報圧縮装置

(57)【要約】
【目的】　時間情報とイベント情報を有する演奏情報データの性質を利用して、データ量を大きく削減することのできる演奏情報圧縮装置を提供する。
【構成】　楽音を構成する音の発音時間情報と共に音の音程その他を指定するイベントを含む演奏情報に応答し、演奏情報のイベント間の相対時間を分析する時間分析手段１２と、演奏情報に応答し、イベントの種類を分析するイベント分析手段１４と、演奏情報に応答し、所定時間間隔でデータを分割してセグメント化し、各々のセグメントから同一内容のデータを検出する楽曲構造分析手段１６と、時間分析手段とイベント分析手段と楽曲構造分析手段のそれぞれの分析結果に基づいて演奏情報を符号化する符号生成手段１８とを、有する。



【特許請求の範囲】

【請求項１】　楽音を構成する音の発音時間情報と共に音の音程その他を指定するイベントを含む演奏情報に応答し、前記演奏情報のイベント間の相対時間を分析する時間分析手段と、

前記演奏情報に応答し、イベントの種類を分析するイベント分析手段と、

前記演奏情報に応答し、所定時間間隔でデータを分割してセグメント化し、各々のセグメントから同一内容のデータを検出する楽曲構造分析手段と、

前記時間分析手段と前記イベント分析手段と前記楽曲構造分析手段のそれぞれの分析結果に基づいて前記演奏情報を符号化する符号生成手段とを、

有する演奏情報圧縮装置。

【請求項２】　前記時間分析手段がイベント間の相対時間の出現頻度を算出する手段と、出現するイベント間の相対時間の最大公約数を算出する手段とを有する請求項１記載の演奏情報圧縮装置。

【請求項３】　前記イベント分析手段が特定の種類のイベントの出現の頻度を算出する手段を有する請求項１記載の演奏情報圧縮装置。

【請求項４】　前記イベント分析手段が全ての種類のイベントの出現の頻度を算出する手段を有する請求項１記載の演奏情報圧縮装置。

【請求項５】　前記イベント分析手段が特定の種類のイベントが所定数以上連続している部分を検出する手段を有する請求項１記載の演奏情報圧縮装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、演奏情報のデータ量を削減する演奏情報圧縮装置に関する。

【０００２】

【従来の技術】一般に通信カラオケと言われるシステムが登場している。これは従来の光ディスク等の記憶媒体に記録された楽音、歌詩、画像等のデータをリクエストに応じて選択して再生するものに代えて、ステーション（ホスト）と各端末を通信回線で結び、新譜を短時間で端末に送る等の配信を可能としたものである。かかる通信カラオケでは、ＭＩＤＩ信号のデータが用いられ、楽音信号は楽音を構成する音の音程と発音時間等を指定する演奏情報として送信されている。従来、ＭＩＤＩデータの保存形式として、スタンダード　ＭＩＤＩ　ファイル（以下ＳＭＦとする）と呼ばれる形式が一般に用いられている。ＳＭＦの詳細は省略するが、個々の演奏情報は、図２に示すように、デルタタイムとイベントという２つの部分で構成される。デルタタイム（Δｔ）は、隣り合ったイベント間の時間間隔を表している。イベントは、音程や音の強さといった種々の演奏情報を含んでいる。この形式は、記憶容量の効率的利用という点では、必ずしも適当なものではない。この理由は大きく分けて

２つある。

【０００３】１つは、イベントデータとして、ＭＩＤＩメッセージをそのまま使っている点である。例えば、ＭＩＤＩメッセージの中に、ある特定の音を止める「ノートオフ」メッセージというものがある。これは、ノートオフであることを示すステータスに１バイト、ノートナンバ（音程）を表すのに１バイト、ベロシティ（鍵盤を離す速さ）に１バイトの合計３バイトで構成される。しかしノートオフのベロシティは実際には一定値であることがほとんどで、この場合１つのノートオフメッセージにつき１バイト無駄な情報が保存されていることになる。

【０００４】もう１つは、１つの楽曲の中で同じパターンが繰り返しある場合でも、繰り返し情報を記述しないで、同じデータを重複して記録している点である。通常の楽曲であれば、同じパターンが繰り返される確率は非常に高いので、重複するデータを省くことができれば、記憶容量の有効利用を図ることができる。なお、音符、記号よりなるメロディの楽譜データについて繰り返し部分を効率よく処理する技術が特公昭６１－９１６９７号公報に示されているが、これはＭＩＤＩ信号を前提とするものではなく、通信カラオケに応用することはできない。さらに特開平４－１４７１９２号公報には楽譜の繰り返し部分をパターン化する技術が示されているが、十分なデータの圧縮は期待できない。

【０００５】

【発明が解決しようとする課題】ＳＭＦには以上のような問題点があるので、特にＭＩＤＩデータを大量に保存する音楽データベースにおいて、記憶容量を削減できる保存形式への要求が高かった。また、通信カラオケ等ではデータ量の削減による通信時間及び通信コストの削減が望まれていた。したがって、本発明は時間情報とイベント情報を有する演奏情報データの性質を利用して、データ量を大きく削減することのできる演奏情報圧縮装置を提供することを目的とする。

【０００６】

【課題を解決するための手段】上記目的を達成するために、本発明では演奏情報に含まれる時間情報及びイベント情報の中から同一の情報の出現頻度を検出して符号化するようにしている。すなわち、本発明によれば、楽音を構成する音の発音時間情報と共に音の音程その他を指定するイベントを含む演奏情報に応答し、前記演奏情報のイベント間の相対時間を分析する時間分析手段と、前記演奏情報に応答し、イベントの種類を分析するイベント分析手段と、前記演奏情報に応答し、所定時間間隔でデータを分割してセグメント化し、各々のセグメントから同一内容のデータを検出する楽曲構造分析手段と、前記時間分析手段と前記イベント分析手段と前記楽曲構造分析手段のそれぞれの分析結果に基づいて前記演奏情報を符号化する符号生成手段とを、有する演奏情報圧縮装

置が提供される。

【０００７】

【実施例】以下、図面と共に本発明の演奏情報圧縮装置の好ましい実施例について説明する。図１はかかる実施例のブロック図であり、この演奏情報圧縮装置はトラック分離手段１０と時間分析手段１２とイベント分析手段１４と楽曲構造分析手段１６と符号生成手段１８で構成される。これらの手段は全体を１つ又は複数のＣＰＵ（中央演算処理装置）で構成することができる。入力データはＳＭＦであり、前述したように個々の演奏情報は、デルタタイムとイベントの２つの部分で構成されている。一般に入力データには複数のＭＩＤＩチャンネルの演奏情報が含まれている。まずトラック分離手段１０において、１トラックに１つのＭＩＤＩチャンネルの演奏情報のみ含まれるように、入力データを分離する。入力データがＳＭＦのフォーマット１のようにあらかじめこの条件を満たしている場合は、この処理を省略することができる。

【０００８】時間分析手段１２はデルタタイムを分析するもので、ここでは入力データに含まれるデルタタイムの種類とその使用頻度を分析する。デルタタイムは、隣り合ったＭＩＤＩイベントの間隔をある時間単位で表したものである。現在市販されているＭＩＤＩデータ作成ソフトにおいては、４部音符の４８０分の１を基本単位（１ｔｉｃｋ）とするものが多いので、以下では４部音符の４８０分の１を単位とした場合を例にして説明する。

【０００９】図３は、音符の例を示すもので、このような音符の並びをＳＭＦで表した場合、デルタタイム及びデルタタイムの記述に必要なバイト数は表１のようになる。時間分析手段１２においては、トラック内のデルタタイムの使用頻度（あるいは、全トラックを通してのデルタタイムの使用頻度）を調べ、表２に示すように、使用頻度の高い順に表２のようなインデックスをつけたデルタタイム頻度表を作り、これを符号生成手段１８に出力する。これは非常に単純な例であるが、通常の楽曲においても使用頻度の高いデルタタイムの数は限られたものになる。使用頻度の高いデルタタイムを短い符号に変換すれば、データ量が削減できる。

【００１０】

【表１】

デルタタイム	イベント	バイト数
0	ノートオン	1
480	ノートオフ	2
0	ノートオン	1
240	ノートオフ	2
0	ノートオン	1
240	ノートオフ	2
0	ノートオン	1
240	ノートオフ	2
0	ノートオン	1
120	ノートオフ	1
0	ノートオン	1
120	ノートオン	1

【００１１】

【表２】

インデックス	デルタタイム	使用回数
0	0	6
1	240	3
2	120	2
3	480	1

【００１２】ＳＭＦにおいてデルタタイムは可変長符号で表され、値が小さいほど必要なバイト数は少なくて済む。デルタタイムの基本単位が細かいほど音楽的な表現力は高いと言えるが、それに従って必要なバイト数も増える。一方実際に楽曲に使われているデルタタイムを調べると、基本単位の１刻み（１ｔｉｃｋ）まで使っていない場合がある。この例では、１２０ｔｉｃｋが実際に使用されている最小単位である。そこでデルタタイムの最大公約数を求め、符号生成手段１８において、個々のデルタタイムをこの最大公約数で除算して符号化する。

【００１３】次にイベント分析手段１４について説明する。イベント分析の処理としては、全てのイベントについて使用頻度を算出する場合と、特に使用頻度の高い「ノートオン」（ある音を出す）、「ノートオフ」（ある音を止める）イベントに絞って使用頻度を算出する場合の２つがあるが、以下ではノートオンとノートオフに絞った場合を中心に説明する。

【００１４】ＳＭＦにおいては図４に示すように、ノートオンイベントを３バイト、ノートオフイベントを３バイトで表している。各々のパラメータは、ノートナンバ（音程）とベロシティ（強さ）の２バイトである。

【００１５】１曲の中で使われるノートナンバとベロシティの組み合わせは、ある程度限られており、しかも同じ組み合わせが繰り返して使われることが多い。したがって、ノートナンバとベロシティの組み合わせの使用頻度を調べ、多く使われるものを短い符号に変換すれば、データ量が削減できる。

【００１６】イベント分析手段１４においては、図５に示す手順でノートオン、ノートオフ各々について、ノートナンバとベロシティの組み合わせの使用頻度をトラック単位あるいは全トラックを通して算出し、使用頻度の高い順にインデックスをつけたノートオン頻度表とノートオフ頻度表をトラック毎にあるいは、全トラックについて１つ作成し、符号生成手段１８に出力する。表３に図３の音符を例とした場合のノートオン頻度表を示す。＊＊ノートオフ頻度表も同様である。以上の実施例では、ノートオンとノートオフのみを対象として、各々の使用頻度を算出したが、全ての種類のイベントを対象にした使用頻度を算出する場合は、ノートオン頻度表、ノートオフ頻度表の代わりに表４のようなイベント頻度表を作ればよい。

【００１７】

【表３】

インデックス	ノートナンバ	ベロシティ	使用回数
0	60	96	3
1	64	96	2
2	64	80	1

【００１８】

※ ※【表４】

インデックス	使用頻度	データ長	データ
0			
1			
.....			
N			

【００１９】ところで、音程を連続的に変えるイベントであるピッチベンドなどは、ＳＭＦにおいてパラメータが少しずつ変わりながら連続して使われていることが多い。このような部分を以下では、連続イベント部分と呼ぶが、連続イベント部分を検出し、その最初と最後の位置を記録して、符号生成手段１８に出力する。この情報の利用については後述する。

【００２０】次に楽曲構造分析手段１６について説明する。まず演奏情報を一定の時間間隔（例えば、１小節相当）で分割する。分割された要素をここではセグメントと呼ぶ。前述したように、通常の楽曲では、同じ内容のセグメントが繰り返して使われている場合が多いので、このようなセグメントの検出を行う。

★【００２１】セグメントには、楽曲の先頭から順に０，１，２…Ｎといった番号を付け、セグメントの先頭位置及び長さの情報と共にセグメント表として記述する。先頭位置は入力データの先頭からのオフセットバイトで表し、長さはセグメントのバイト数で表す。ｉ番目のセグメントに対し、０～ｉ－１番目のセグメントとの比較を行い、同じ内容のセグメントがあれば、最も若い番号をセグメント表に記録する。このような処理をｉ＝１～Ｎとして行った後、セグメント表を符号生成手段１８に出力する。セグメント表の一例を表５に示す。

【００２２】

【表５】

★

セグメント番号	先頭位置	長さ	同じ内容のセグメント
0	0	100	なし
1	100	120	なし
2	220	100	1
3	320	120	2
4	440	100	なし
5	540	120	2
6	660	120	5

【００２３】次に符号生成手段１８について説明する。本装置で生成する符号は図６の（ａ）に示すように、ヘッダ、デルタタイム、ノートオンマップ、ノートオフマップ、データブロックの各部分で構成されている。データブロックの要素は、図７に示すように、ｍビットのデルタタイムフラグとｎビットのイベントフラグと可変長のデータ部で構成される。

【００２４】符号生成手段１８では、トラック分離手段

１０の出力を順次走査し、デルタタイムとイベントを順次読み込む。その際、楽曲構造分析手段１６で作成されたセグメント表を参照して、現在読み込んでいるデータがどのセグメントに属しているか、またそのセグメントと同じセグメントの有無についてチェックする。セグメント表に同じ内容のセグメントがある場合は、イベントフラグにセグメント参照を意味するコードをセットし、データ部に参照するセグメント番号を出力する。そして次のセグメントが始まるまでトラック分離手段１０から出力を読み飛ばす。セグメント表に同じ内容のセグメントが無い場合は、イベントフラグにセグメント定義を意味するコードをセットする。そのセグメントについて以下の処理を行った後、セグメント定義終了を意味するコードをイベントフラグにセットする。以上セグメント記述をまとめると図１０のようになる。

【００２５】セグメントを定義する場合の処理は次のようになる。デルタタイムの値が、前記デルタタイム頻度表の先頭から2^m-1個の範囲にあれば、デルタタイムフラグに、そのインデクッスを記述する。この場合、データ部にはデルタタイムのデータは出力しない。またこの範囲になければ、ｍビットを全て１にして、データ部には時間分析手段１２で求めた最大公約数で除したデルタタイムをＳＭＦと同様な可変長符号で記述する。

【００２６】イベントフラグはｎビットあるので、2^n種類のイベントが記述できるが、マップにないことを示すのに１個、セグメントの記述用に３個必要なので、ノートオン用にｐ個、ノートオフ用にｑ個、連続イベント記述用にｒ個のコードを割り当てる（ただし、$p+q+r<2^n-4$）。イベントがノートオンである場合は、ノートオン表を参照し、インデックスが０からｐ－１の範囲で、ノートナンバとベロシティが一致すれば、そのインデックスをイベントフラグにセットする。この場合は、データ部に出力しない。もしノートオン表のその範囲になければ、ｎビットを全て１にして、データ部にＳＭＦと同様のイベント情報を記述する。ノートオフについても同様である。

【００２７】イベント分析手段１４で検出した連続イベント部分を読み込んだ場合、以下の処理を行う。イベントフラグには、連続イベント用に割り当てられたｒ個のコードの中から、イベントの種類に対応したものを選び、それを設定する。データ部は、図８の（ａ）で示したフォーマットとする。

【００２８】図９に示した連続イベントを例に取って、具体的に説明する。連続イベント部分は、この例のようにほぼ同じ時間間隔でパラメータ値が少しずつ変化していることが多い。この例では、ピッチベンドイベントが６個あり、イベントの時間間隔は全て５０、パラメータ値は各々１０、１１、１２、１５、１８、２１となっているものとする。ＳＭＦで記述する場合は、各々のデルタタイムに１バイト、イベントに３バイト必要として、６×（１＋３）＝２４バイト必要である。この例を図８の（ａ）に従って、実際に記述したものが図８の（ｂ）である。８バイトで記述ができ、ＳＭＦに比べて大幅に記憶容量が削減できる。

【００２９】ヘッダには、図６の（ｂ）に示すように、トラックの長さ、デルタタイムの最大公約数が記述される。デルタタイムマップは図６の（ｃ）に示すように、マップに含まれるデルタタイムの数と、時間分析手段１２で作成したデルタタイム頻度表の中のインデックスが０から（2^m-2）を超えない範囲のデルタタイムを順番に並べたものである。

【００３０】ノートオンマップは図６の（ｄ）に示すように、マップに含まれるデータ数と、イベント分析手段１４で作成したノートオン頻度表のインデックスが０からｐ－１までのノートナンバとベロシティを順番に並べたものである。ノートオフマップも同様であるが、ノートオンとは違いノートオフのベロシティは全て同じ値になっていることが多いので、ベロシティ情報を記述する１バイトを追加する。ノートオフベロシティが全て同じ値であれば図６の（ｅ）に示すように、その値（０～１２７）をこの場所に記述し、続いてノートオフ頻度表のノートナンバのみｑ個並べる。もし一定値でなければ、図６の（ｆ）に示すように先頭の１バイトに１２８以上の適当な値を書き込み、続いてノートナンバとベロシティの組をｑ個並べる。なお、この例では、デルタタイムマップ、ノートオンマップ、ノートオフマップをトラック毎に作成したが、これらを全トラックに対して１つずつ用意してもよい。

【００３１】以上本発明の演奏情報圧縮装置について説明したが、図３の音符の並びが具体的にどのような符号に変換され、どの程度圧縮されるかを示す。ノートオフベロシティは全て６４で一定とする。ＳＭＦで記述する場合、トラックの長さを表すのに４バイト、表１に示したようにデルタタイムに１６バイト、ノートオン６回、ノートオフ６回各々３バイトずつとして３６バイト、合計５６バイト必要である。

【００３２】一方ｍ＝３、ｎ＝５、ｐ＝３、ｑ＝２とした場合の本装置の出力はヘッダ５バイト、デルタタイムマップ５バイト、ノートオンマップ７バイト、ノートオフマップ４バイト、データブロック１２バイトで合計３３バイトとなり、データ圧縮される。この例はデータ数が少なく、セグメントの重複が無い例であるが、一般の楽曲ではセグメントの重複があるので更に効率的に圧縮できる。

【００３３】ノートオン頻度表、ノートオフ頻度表の代わりにイベント頻度表を使う場合は、出力は、図６の（ｇ）に示すような構成となる。イベントマップは図６の（ｈ）に示すように、イベントマップに含まれるデータの個数に続き、イベント頻度表のデータ長とデータの組を2^n-1個超えない範囲で並べる。また、データブ

ロックのイベント記述は、2^n-1個のコードを使って上記と同様に行なえば良い。

【００３４】なお、以上詳述したフォーマット並びに処理手順は一例であり、その主旨を逸脱しない範囲において種々の変更を加えることができる。

【００３５】

【発明の効果】以上説明したように本発明の演奏情報圧縮装置は上記構成なので、原データの持つ情報を全く失うことなく、大幅なデータ削減が可能であり、特にＭＩＤＩデータを大量に保存する音楽データベースにおいては、記憶容量が少なくてすみコストが削減できる。またＭＩＤＩデータを通信回線で送る場合には、送信時間が短縮でき、コストも削減することができる。

【図面の簡単な説明】

【図１】本発明の演奏情報圧縮装置の好ましい実施例のブロック図である。

【図２】ＳＦＭデータの説明図である。

【図３】演奏データの一例を示す図である。

【図４】ＳＭＦのノートオン、ノートオフの構成図である。

【図５】イベント分析手段の動作を示すフローチャートである。

【図６】本装置の出力する符号の構成を示す図である。

【図７】データブロックの要素の説明図である。

【図８】連続イベントを記述するフォーマットを示す図である。

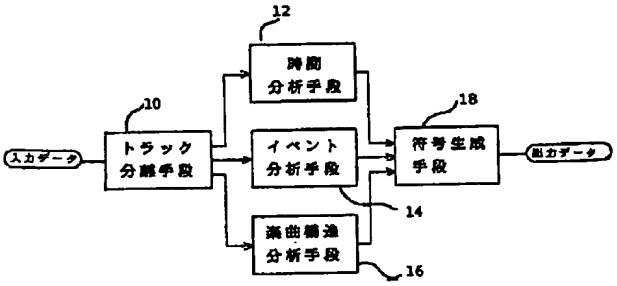
【図９】連続イベントの一例を示す図である。

【図１０】セグメントを記述するフォーマットを示す図である。

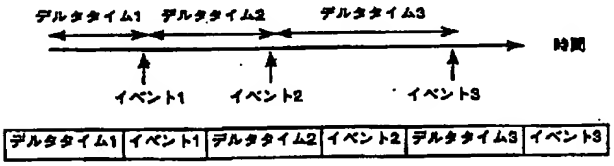
【符号の説明】

１０　トラック分離手段
１２　時間分析手段
１４　イベント分析手段
１６　楽曲構造分析手段
１８　符号生成手段

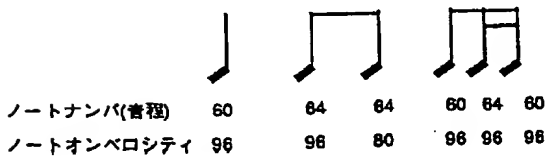
【図１】



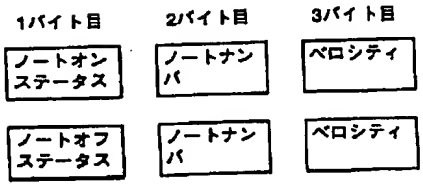
【図２】



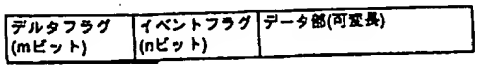
【図３】



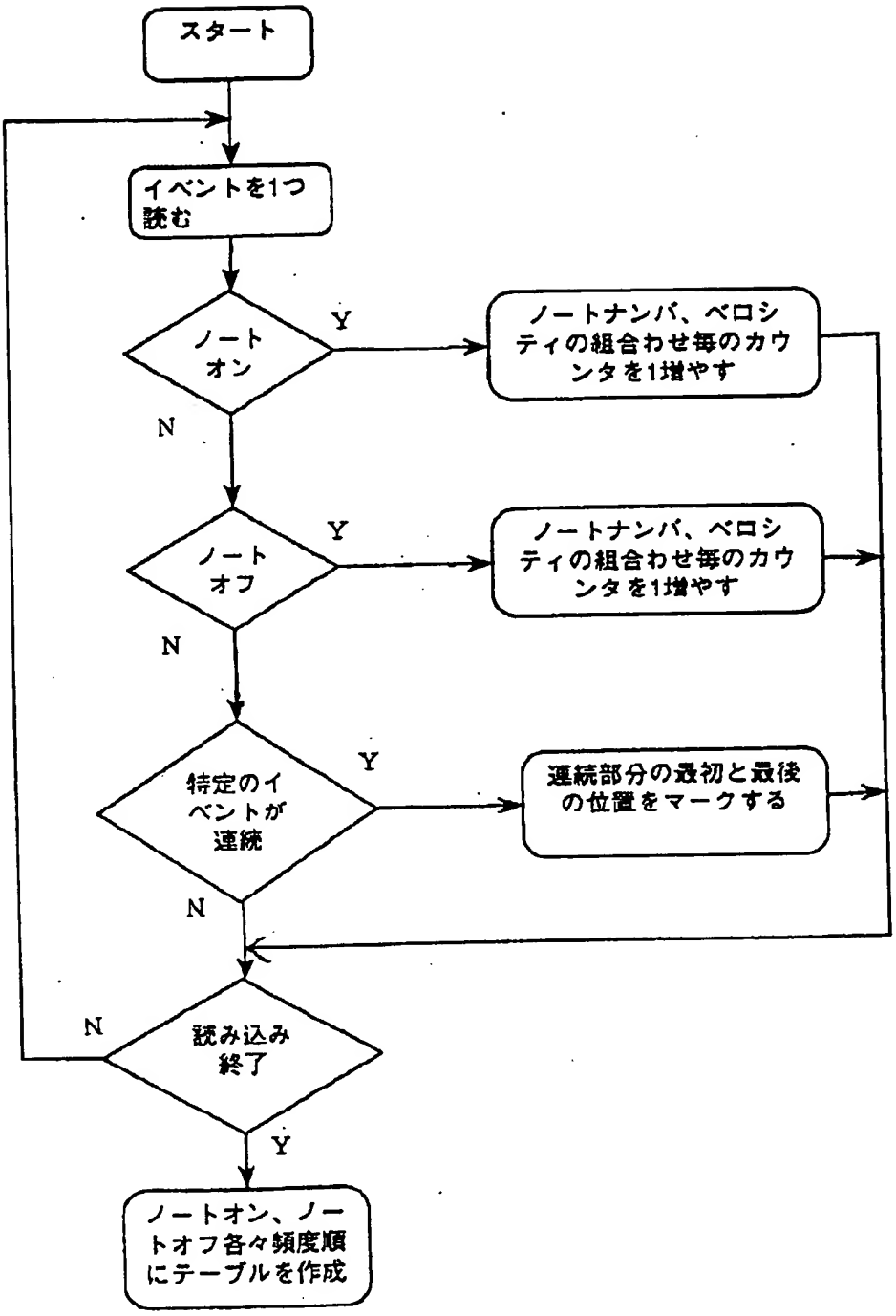
【図４】



【図７】



【図５】


スタート
イベントを1つ読む
ノートオン
Y
ノートナンバ、ベロシティの組合わせ毎のカウンタを1増やす
N
ノートオフ
Y
ノートナンバ、ベロシティの組合わせ毎のカウンタを1増やす
N
特定のイベントが連続
Y
連続部分の最初と最後の位置をマークする
N
読み込み終了
N
Y
ノートオン、ノートオフ各々頻度順にテーブルを作成

【図６】

(a)

ヘッダ
デルタタイムマップ
ノートオンマップ
ノートオフマップ
データブロック

(g)

ヘッダ
デルタタイムマップ
イベントマップ
データブロック

(b)

トラックの長さ(4)	デルタタイムの最大公約数(1)

(c)

デルタタイムマップの個数(1)	デルタタイム0(可変長)	デルタタイム1(可変長)

(d)

ノートオンマップの個数(1)	ノートナンバ0(1)	ベロシティ0(1)	ノートナンバ1(1)	ベロシティ1(1)

(e)

ベロシティ64	ノートオンマップの個数(1)	ノートナンバ0(1)	ノートナンバ1(1)

(f)

ベロシティ64	ノートオンマップの個数(1)	ノートナンバ0(1)	ベロシティ0(1)	ノートナンバ1(1)	ベロシティ1(1)

(h)

イベントマップの個数(1)	イベント0の長さ	イベント0のデータ	イベント1の長さ	イベント1のデータ

【図８】

(a)

直線部分の数(1)	初期値(1~2)	時間間隔(可変長)	前の値との差分(1~2)	繰り返し数(1)

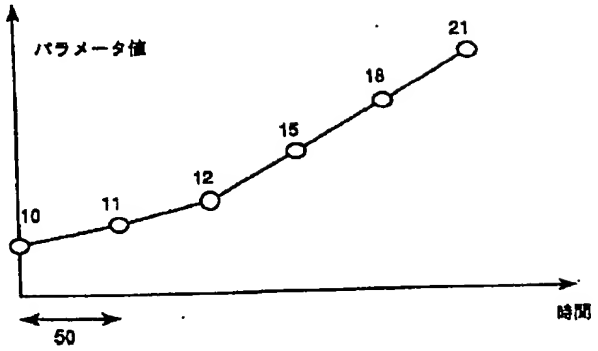
(1)は1バイトを表す

必要な回数繰り返す

(b)

2	10	50	1	2
		50	3	3

【図９】



【図１０】

同じ内容のセグメントを参照する場合

デルタタイムフラグ	イベントフラグ	データ部
	セグメント参照コード	セグメント番号

セグメントを定義する場合

デルタタイムフラグ	イベントフラグ		デルタタイムフラグ	イベントフラグ
	セグメント定義コード	そのセグメントの符号		定義終了コード